2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Regrasp Planning of Three-Fingered Hand for a Polygonal Object

Thanathorn Phoka and Attawith Sudsang

Abstract— **This paper addresses the problem of regrasp planning for a polygon with a large number of edges. We propose an approach for computing sequences of finger repositioning that allow the hand to switch from one grasping configuration to another while maintaining force-closure during the entire process. The proposed approach is based on exploring a structure called switching graph. Complete sets of two-fingered force-closure grasps are computed in grasp space. Adjacent sets of force-closure grasps are merged into a connected set which allows finger repositioning by continuous movements of fingers on adjacent polygonal edges. We present an output sensitive algorithm to construct a switching graph from the obtained connected sets. A method for finding the optimal solution of a finger switching is also presented. The proposed approach has been implemented and some preliminary results are presented.**

I. Introduction

Regrasp planning is inspired by the actual behavior of human manipulation when the object goes through several different grasping configurations while being kept in the hand during the entire process. There are several complex tasks that involve not one grasping configuration but a sequence of different grasping configurations, or it might happen that the current grasp is not suitable for the task to perform. This arises the *regrasp planning problem*, given an initial grasp and a desired grasp, the goal is to compute a movement sequence of the fingers' contact points that changes the initial grasp to the desired configuration while still maintaining force-closure.

Changing grasping configuration while maintaining force-closure was suggested by [1]. Several simple actions can change grasping configuration without losing force-closure. For example, consider sliding and rolling [2] of finger. Finger gaiting [3] or finger switching [4], on the other hand, involves placing one additional end effector on a new position and removing one of the initial end effector.

In this work, we assume that an object is modeled by a polygon with a large number of edges to achieve a fine level-of-detail of the object. The regrasp planning problem in this setting remains mostly unexplored. Regrasp planning for discrete contact points using independent regions is proposed in [5]. The regrasp operation that is allowed in the work is only motion of a finger without contact breaking. The main restriction of applying only regrasp operation is that the approach fails to find a path between two grasps in distinct connected grasp sets. To overwhelm this limitation, we allow the finger switching operation to change grasping configurations between two disconnected grasp sets. We consider the case that uses minimum number of fingers to grasp and regrasp, i.e., two-fingered grasps is taken into account and one additional finger is used for a finger switching. Therefore, a hand applied in this work is assumed to equip with three fingers. To obtain the complete structure for regrasp planning of an object, all force-closure grasps are computed in grasp space. Instead of naively constructing a switching graph from all uncombined sets of grasps computed from all combination of polygonal edges, we propose to merge sets of grasps that connect to one another into a connected set. In grasp space, a connected set allows continuous changing among any grasping configurations in the set, i.e., it allows continuous movements of fingers across polygonal edges. The obtained connected sets are then used to construct nodes of a switching graph. We also propose an output sensitive algorithm which efficiently computes all edges of a switching graph. Regrasp planning then can be formulated as a graph search problem where nodes of the graph represent connected sets of force-closure grasps while an edge connects two sets that can be changed between each other by finger switching.

This research is financially supported in part by the Thailand Research Fund through the Royal Golden Jubilee Ph.D. program under grant No. Ph.D. 1.O.CU/49/D.1 and the 90^{th} Anniversary of Chulalongkorn University Fund through the Ratchadapiseksomphot Fund, both are greatly appreciated.

T. Phoka and A. Sudsang are with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 10330, Thailand thanathorn.p@gmail.com, attawith@gmail.com

II. Background

In 2D, a hard finger in contact with some object at a point $\mathbf{x} = (x_1, x_2)$ exerts a force $\mathbf{f} = (f_1, f_2)$ parallel to the normal vector of the contact surface. In the presence of friction, a single contact point can exert forces in different directions. Under the Coulomb friction model, the angle between a force exerted by the finger and the inward pointing normal cannot be greater than $\theta = \tan^{-1}\mu$ where μ denotes the given frictional coefficient.

It is formally shown in [6] and [7] for two finger cases that a sufficient condition for force-closure is non-marginal equilibrium grasps, i.e., grasps such that the forces achiev-

ing equilibrium lie strictly inside the friction cones at the fingertips.

Proposition 1: A sufficient condition for two-fingered force-closure is non-marginal equilibrium

That is, grasps achieving equilibrium with non-zero forces for some friction coefficient achieve force-closure for any strictly greater friction coefficient. Due to [6], the following proposition characterizes two-fingered equilibrium.

Proposition 2: A necessary and sufficient condition for two points to form an equilibrium grasp with non-zero contact forces is that the line joining both points lies completely in the two double-sided friction cones at the points.

III. Regrasp Planning

Before describing how our algorithm can solve regrasp planning problem, it is needed to understand how the regrasp can be achieved as a sequence of finger repositioning. In this paper, we classify finger repositioning into two different types: *finger switching* and *finger aligning*. Given an object initially in a force-closure grasp, a finger switching is a process that a free finger is appropriately placed on the grasped object to form another force-closure grasp, so that the finger in the original grasp that is not involved in the new grasp can be lifted off allowing the hand to change from the original grasping configuration to the new one while maintaining that the object stays in a force-closure grasp during the finger swapping. Of course, the fingers in the original grasp are sometimes not in the positions where a finger switching can take place. They have to be appropriately repositioned to allow the intended finger switching. This situation calls for a finger aligning. Finger aligning refers to a process that grasping fingers, without breaking contact with the object, adjust their contact positions while maintaining a force-closure grasp during the entire process. With frictional contact assumed, this may be implemented using finger sliding or rolling [2].

To determine an appropriate sequence of these two processes, we introduce a structure called a switching graph. A node in a switching graph represents a connected set of force-closure grasps. An edge connecting two nodes indicates that there exists a grasp associated with one node that can be switched to a grasp associated with the other by finger switching. By using a switching graph, the regrasp problem can be formulated into a graph search problem. A path from the graph search determines a sequence of actions – switching and aligning to be executed in order to traverse from the initial to the final grasp. The following sections will describe representation of grasp sets, the finger switching and aligning primitives and the switching graph in detail.

IV. Representing force-closure grasps

In this section, we describe how to represent and construct the configuration space that characterizes all force-closure grasps. The full description from [8] is briefly explained here. The object of interest is assumed to be simple. The configuration of the problem consists of two parameters, each of which defines where a finger is placed along the boundary of the grasped object.

We now define entities of a polygon needed in our consideration as follows. A simple polygon P is defined by n distinct vertices $\mathbf{v}_i \in \Re^2$ where $i \in \mathbf{Z}_n$ [1]. We will assume that $\mathbf{v}_i$ are arranged counterclockwise. Edges E_i are line segments with endpoints $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$. Every point $\mathbf{p}$ on P's boundary can be mapped to the length of curve measured counterclockwise from $\mathbf{v}_0$ to $\mathbf{p}$ along the boundary. We will write $length(\mathbf{p})$ to represent such length. Lengths of E_i can be computed by the equation $l_i = \|\mathbf{v}_{i+1} - \mathbf{v}_i\|$. It is obvious that $L_i = length(\mathbf{v}_i) = \sum_{j \in \mathbf{Z}_i} l_j$. We denote by L the total length of P's boundary, which can be computed by $L = \sum_{i \in \mathbf{Z}_n} l_i$.

Next, let us define tangents of E_i as $\mathbf{t}_i = (\mathbf{v}_{i+1} - \mathbf{v}_i)/l_i$. The normal vectors $\mathbf{n}_i$ of E_i are unit vectors that are perpendicular to $\mathbf{t}_i$ and point inward ($\mathbf{n}_i$ can be obtained by rotating $\mathbf{t}_i$ $\pi/2$ radian counterclockwise). The cone of forces C_i that can be exerted on the edge E_i is defined by two vectors $\mathbf{n}_i + (\tan\alpha)\mathbf{t}_i$ and $\mathbf{n}_i - (\tan\alpha)\mathbf{t}_i$ where $\alpha \in [0, \pi/2)$ is the half-angle of the friction cone.

The configuration space or grasp space $\mathbf{C}$ of the two fingers is $[0, L) \times [0, L)$. Given a configuration $(u, u') \in \mathbf{C}$, we say that (u, u') composes a two-fingered grasp if and only if the two contact points $length^{-1}(u)$ and $(length)^{-1}(u')$ produce force-closure. (Recall that $length$ is a function that maps a vertex into a number, so $length^{-1}$ gives a vertex.) The *grasp set* $G \subseteq \mathbf{C}$ is the set of all configurations that compose two-fingered grasps. *Grasp subsets* $G_{i,j}$ are graspable regions on edges E_i and E_j defined by

$$G_{i,j} = G \cap ([L_i, L_{i+1}] \times [L_j, L_{j+1}]).$$

A. Computing $G_{i,j}$

Because each $G_{i,j}$ corresponds to configurations whose one finger is on E_i and the other is on E_j, $G_{i,j}$ has been well-studied. According to Proposition 2, it has been shown in [9] that $G_{i,j}$ can be defined by eight linear inequalities in the parameters u and u'. However, there is an easier way to define $G_{i,j}$ as follows. Define the inverted force cone $-C_j$ as $\{-\mathbf{x} \mid \mathbf{x} \in C_j\}$. [6] showed that emptiness of $C_{i,j} = C_i \cap (-C_j)$ implies emptiness of $G_{i,j}$. If $C_{i,j}$ is not empty, we claim that $G_{i,j}$ can be defined by no more than six points on the bounding rectangle.

Since a two-fingered grasp can be either *compressive*(squeezing grasp) or *expansive*(stretching grasp), we define for simplicity $DC_{i,j} = C_{i,j} \cup (-C_{i,j})$ as the double-sided cone of $C_{i,j}$ so that both the stretching and squeezing cases can be dealt with together. Let $DC_{i,j}(\mathbf{v})$ be the cone $DC_{i,j}$

[1] $\mathbf{Z}_n$ is a group of non-negative integers less than n. Addition and subtraction are computed modulo n.

centered at $\mathbf{v}$. All defining points of $G_{i,j}$ can be found by computing endpoints of four intersections: $DC_{i,j}(\mathbf{v}_i) \cap E'_j$, $DC_{i,j}(\mathbf{v}_{i+1}) \cap E'_j$, $DC_{i,j}(\mathbf{v}'_j) \cap E_i$, and $DC_{i,j}(\mathbf{v}'_{j+1}) \cap E_i$.

B. Constructing G

Now we know that each $G_{i,j}$ contains at most six defining vertices, so all $G_{i,j}$ can be constructed within $O(n^2)$ time. With all $G_{i,j}$ in hand, we can construct a switching graph by defining each $G_{i,j}$ for a node in the graph but we can do better by exploiting connectivity among adjacent $G_{i,j}$. This allows us to continuously change configurations between any two grasps in adjacent $G_{i,j}$. In our algorithm, we will need the polygonal representation of G, so adjacent $G_{i,j}$ must be merged together into big pieces. The detail of the algorithm described in [8] for construction of G from all $G_{i,j}$ is neglected in this paper. It spends $O(n^2)$ time to merge all connected polygons. Let m be the number of connected polygons describing G and all polygons be defined by P_a where $a = 1 \dots m$. A polygon represents a connected set of two-fingered grasps on connected edges and involve with a node in the switching graph. All polygons P_a, $a = 1 \dots m$ are contained in nodes v_a of the switching graph.

V. Finger Switching

Regrasp process which changes grasping configuration by placing an additional finger on desired contact point and then releasing one finger of the initial grasp is called finger switching. Intuitively, considering grasps on two different grasp sets, a finger switching can be performed when there exists a common contact point on the grasped object. In grasp space, the common contact points are computed in subspaces of one parameter. It requires projections of two grasp sets onto the subspaces. The projections are then checked for the intersection which indicates a set of common contact points. This operation involves with an edge in the switching graph. Considering two grasp sets associated with two nodes, existence of finger switching between these sets indicates an edge linking the two nodes.

Finger switching requires that one non-switching contact points must remain the same during the process. Formally, there will be an edge connecting a node v_a and a node v_b when there exists a couple of points $(length^{-1}(u_a), length^{-1}(u'_a))$ where $(u_a, u'_a) \in P_a$ and a couple $(length^{-1}(u_b), length^{-1}(u'_b))$ where $(u_b, u'_b) \in P_b$ such that $u_a = u_b$ or $u'_a = u'_b$.

To check whether there exist grasps from two grasp sets that can switch to each other, we consider two polygons representing these grasp sets. The projection $\pi_u(P_a)$ of P_a on the axis of parameter u_a (Fig. 1(a)) represents the set of points on the object that are possible to form two-fingered grasps with some points corresponding to the projection $\pi_{u'}(P_a)$ of P_a on the axis of parameter u'_a. Similarly, the projections of P_b (Fig. 1(b)) represents the subspaces of two-fingered grasps . Note that $\pi_u(P_a)$ and $\pi_u(P_b)$ are in the same subspace, if the intersection between these two projections is not empty (Fig. 1(c)), then there exists points $length^{-1}(u_a)$ on the object that form two-fingered grasps with $length^{-1}(u'_a)$ and $length^{-1}(u'_b)$ concurrently when $u_a = u_b$ is satisfied. We apply the same process to check the existence of finger switching on the space of parameters u'_a and u'_b by considering $\pi_{u'}(P_a)$ and $\pi_{u'}(P_b)$.

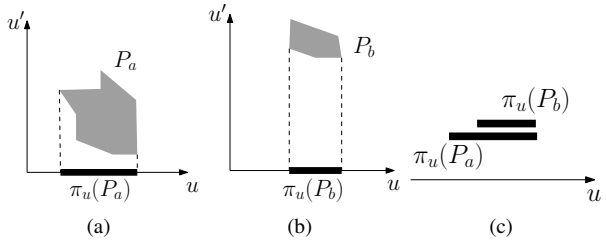

Fig. 1. Finger switching : The projection of (a) P_a, (b) P_b on u parameter space. (c) Overlapping projections

VI. Finger Aligning

Finger aligning is a process for repositioning fingers by rolling or sliding them along adjacent edges of a polygon while maintaining a force-closure grasp during the repositioning process. By applying this operation, we can continuously change grasping configurations within the same connected set of grasps. This expresses the direct relation between finger aligning and a node of the switching graph. Each node in the switching graph corresponds to exactly one connected grasp set. Every grasp in each node can be repositioned to another grasp of the same node by finger sliding or rolling because of connectivity in the connected set of two-fingered grasps.

VII. Constructing Switching Graph

To construct a switching graph, all of its vertices and edges have to be found. We compute all $G_{i,j}$ and construct G to identify all connected polygons $P_1, \dots, P_m$. Each connected polygon is associated with a node of the switching graph.

To construct all edges, all polygons have to be checked for finger switching among them. Instead of exhaustively testing all pair of polygons, we apply a sweep algorithm to find overlaps among the projections of the polygons in one parameter subspace (either u or u' in a time). Let the projection $\pi_u(P_a)$ of a polygon P_a on the axis of parameter u_a be represented by an interval $(l_a : h_a)$ where l_a is the lower endpoint and h_a is the higher endpoint. The lower endpoint and the higher endpoint are obtained from the leftmost vertex and the rightmost vertex of P_a respectively. The intervals of all polygons are used in our algorithm. We firstly sort all endpoints in increasing order and store the sorted endpoints into an event queue E. A priority queue Q is used to store intervals and identify overlaps among intervals. The priority of a interval is based on the value of its higher endpoint, less value has more priority. We are now ready

to start the sweeping process from the first element in E. An endpoint is dequeue from E. If it is a lower endpoint, its associated interval is added into Q. Otherwise, if it is a higher endpoint, its associated interval is dequeued from Q. Clearly, the interval has the highest priority, we can use *ExtractMin* operation of the priority queue which removes the highest priority element of the queue to remove the interval from Q. Let the dequeued interval be $(l : h)$. All remaining intervals in Q have the lower endpoints which are less than h and the higher endpoints which are higher than h. Therefore, they overlap the interval $(l : h)$. As a consequence, the associated node of the interval $(l : h)$ has edges linking nodes associated with the remaining intervals in Q. The process is repeated from dequeuing E and so on until E is empty. We also have to check overlaps in the subspace u' using the same algorithm.

The construct of the event queue E takes $O(m \log m)$ running time. A priority queue using a heap give performance $O(1)$ to insert an element into Q. *ExtractMin* operation takes $O(\log m)$. For all endpoints, our output sensitive algorithm takes $O(m \log m + mk)$ where k is the average number of overlapping intervals of one interval.

VIII. Using Switching Graph

A. Unconstrained Regrasp Sequence

A switching graph provides a tool for planning a regrasp sequence. A path connecting the node containing the initial grasping position and the node containing the required grasping position indicates a sequence of edges that a finger switching should be performed. However, a path in a switching graph does not directly indicate which contact points on grasping edges are to be used in each step. For a pair of nodes having an edge connecting them, a switching graph tells us that we can switch between two grasps from two grasp sets but it does not tell which grasping points that we can perform a finger switching. This section describe a method of transforming a path in a switching graph to an actual regrasp sequence.

First, let us consider a finger switching. Finger switching takes place when we move from one node to another node in a graph. An edge in the graph tells us that a finger switching is viable. We have to find two grasps on each node that have one non-switching contact point in common. We pick a point from the intersection of the projections described in section V. That point indicates one actual non-switching point. The next step is to find a point forming a grasp of the first node and a point forming a grasp of the second node. Let us consider a polygon defined in section V. Once a value of u_a or u'_a in the intersection of the projections of P_a and P_b is chosen, we can construct a set of feasible contact points for the other finger by intersecting P_a with the line passing u_a and parallel with the axis of u'_a or the line passing u'_a and parallel with the axis of u_a

Next, let us consider a finger aligning. Finger aligning may be required in-between two finger switchings, i.e., when we just traversed from node v_a to node v_b and about to move to the next node v_c. Let us assume that the first finger switching is just performed and we currently are in a grasp represented in v_b. In order to perform the next finger switching, i.e., to move to the node v_c, the grasping position must have one contact point in common with the final grasp. However, it might not be the current grasp. When an appropriate grasping configuration is computed as described earlier in this section, we have to change from the finishing grasp of the first switching to the a next switching. Since these two grasps are from the same connected set, we can change the current grasp to an appropriate grasp for the next switching by a finger aligning.

B. Optimal Regrasp Sequence

In this section, we will plan for a regrasp sequence that independent contact regions (ICR) are locally optimized for each finger switching using the principle of L_∞ Voronoi diagram [10]. The use of L_∞ Voronoi diagram for optimizing ICR in grasp planning is proposed in [8]. ICR are defined by a rectangle in G whose shorter side length is maximum. We extend to the problem of optimizing ICR for a finger switching which involves two grasps concurrently. The measure of goodness of two rectangles R_a with side lengths a_1 and a_2 and R_b with side lengths b_1 and b_2 is given by $f(R_a, R_b) = \min\{a_1, a_2, b_1, b_2\}$. Our goal is to maximize $f(R_a, R_b)$ such that the grasps represented by the centers of R_a and R_b can switch to each other.

To optimize the criterion, we use another representation of a square to describe ICR. Let $\mathbf{v}$ be a point in G and let $square(\mathbf{v})$ denote the largest square centered at $\mathbf{v}$ that is fully contained in G. The size of $square(\mathbf{v})$ is determined by $size(\mathbf{v}) = min_{\mathbf{p}\in\partial G}(d(\mathbf{v}, \mathbf{p}))$ where $d(\mathbf{v}, \mathbf{p}) = max(|u_\mathbf{v} - u_\mathbf{p}|, |u'_\mathbf{v} - u'_\mathbf{p}|)$. Therefore, any largest square is described by its center $\mathbf{v}$ and $size(\mathbf{v})$. Let $\mathbf{v}_a \in P_a$ and $\mathbf{v}_b \in P_b$, it is clear that maximizing the criterion is equivalent to maximizing $\min\{size(\mathbf{v}_a), size(\mathbf{v}_b)\}$. We denote by $\pi_u(\mathbf{v})$ and $\pi_{u'}(\mathbf{v})$ the projections of a point $\mathbf{v}$ on the axis of u and u' parameters respectively. The problem is transformed to locating the centers $\mathbf{v}_a \in P_a$ and $\mathbf{v}_b \in P_b$ of two squares such that $\pi_u(\mathbf{v}_a) = \pi_u(\mathbf{v}_b)$ or $\pi_{u'}(\mathbf{v}_a) = \pi_{u'}(\mathbf{v}_b)$ and $\min\{size(\mathbf{v}_a), size(\mathbf{v}_b)\}$ is maximal.

Our algorithm exploits an important characterization of Voronoi edges which allows us to search squares centered on them to optimize the criterion. We claim that the largest $square(\mathbf{v})$ must be centered on a Voronoi edge when one parameter of the point $\mathbf{v} \in P$ is restrained. This is justified by the following argument:

We describe in the case that parameter u is restrained as shown by the dotted line in Fig. 2(a).

- If $\mathbf{v}$ is inside a Voronoi region whose upper and lower

boundaries are a Voronoi edge and an edge of P ($\mathbf{v}_4$ in Fig. 2(a)), $square(\mathbf{v})$ must have one corner on that polygonal edge. Moving $\mathbf{v}$ *away* from that polygonal edge will increase $size(\mathbf{v})$. We can move $\mathbf{v}$ in such direction until it reaches a Voronoi edge while $square(\mathbf{v})$ is growing.

- If $\mathbf{v}$ is on a Voronoi region whose upper and lower boundaries are both Voronoi edges ($\mathbf{v}_2$ in Fig. 2(a)), moving $\mathbf{v}$ in one direction, $size(\mathbf{v})$ is increasing or decreasing along the way, and $size(\mathbf{v})$ is decreasing or increasing along the opposite way until it reaches a Voronoi edge ($\mathbf{v}_1$ and $\mathbf{v}_3$ in Fig. 2(a)).

This argument allows us to search two points $\mathbf{v}_a \in VE_a$ and $\mathbf{v}_b \in VE_b$ such that $\pi_u(\mathbf{v}_a) = \pi_u(\mathbf{v}_b)$ or $\pi_{u'}(\mathbf{v}_a) = \pi_{u'}(\mathbf{v}_b)$ for optimizing $\min\{size(\mathbf{v}_a), size(\mathbf{v}_b)\}$ where VE_a and VE_b are sets of Voronoi edges of P_a and P_b.

Searching procedure begins with identifying an interval for finger switching by the intersection between $\pi_u(P_a)$ and $\pi_u(P_b)$ or $\pi_{u'}(P_a)$ and $\pi_{u'}(P_b)$. We again describe in the case of an interval in the space of parameter u. Let $\pi_u(P_a) \cap \pi_u(P_b) \neq \emptyset$ be denoted by an interval $(l : h)$. Voronoi edges in VE_a and VE_b that intersect this interval are considered. It is possible that we obtain many branches of Voronoi edges. All combinations of pairs of Voronoi edge branches are investigated, a pair consists of one Voronoi edge branch from VE_a and another branch from VE_b. For each pair, we divide the two branches using subintervals defined by all Voronoi vertices in the branches as shown by the dotted lines in Fig. 2(b). Voronoi vertices are used to determine subdivisions of two Voronoi edge branches because sizes of largest squares centered at them are critical. Each pair of subsets of two Voronoi edges in a subinterval is then searched for local optimization of the criterion. Let the two subsets be represented by two segments whose endpoints are $\mathbf{s}_a, \mathbf{t}_a$ and $\mathbf{s}_b, \mathbf{t}_b$. Since the size of a square centered on a Voronoi edge linearly increases or decreases and a Voronoi edge is also linear, therefore the size of a square can be parameterized by a parameter α. We define new size functions as

$$size_a(\alpha) = size(\mathbf{s}_a) + \frac{\alpha}{r}(size(\mathbf{t}_a) - size(\mathbf{s}_a))$$

$$size_b(\alpha) = size(\mathbf{s}_b) + \frac{\alpha}{r}(size(\mathbf{t}_b) - size(\mathbf{s}_b))$$

where r is the length of the associated subinterval and $\alpha \in [0, r]$. Clearly, α can be linearly inverted for positions on the segments. Let $\uparrow, \downarrow$ be increasing and decreasing. The locally optimum is obtained as follows.

- If $\alpha \uparrow \Rightarrow size_a(\alpha) \downarrow$ and $size_b(\alpha) \downarrow$, $\alpha = 0$ induces a local optimum.
- If $\alpha \uparrow \Rightarrow size_a(\alpha) \uparrow$ and $size_b(\alpha) \uparrow$, $\alpha = r$ induces a local optimum.
- If $\alpha \uparrow \Rightarrow size_a(\alpha) \downarrow$ and $size_b(\alpha) \uparrow$ and $size_a(0) \leq size_b(0)$, $\alpha = 0$ induces a local optimum.
- If $\alpha \uparrow \Rightarrow size_a(\alpha) \downarrow$ and $size_b(\alpha) \uparrow$ and $size_a(r) \geq size_b(r)$, $\alpha = r$ induces a local optimum.
- If $\alpha \uparrow \Rightarrow size_a(\alpha) \downarrow$ and $size_b(\alpha) \uparrow$ and $size_a(0) > size_b(0)$ and $size_a(r) < size_b(r)$, α causing $size_a(\alpha) = size_b(\alpha)$ induces a local optimum.
- The remaining cases are replica of the last three cases.

All pairs of segments from all subintervals are queried for local optima. Our approach is done when all combinations of pairs of Voronoi edge branches have been explored. The best local optimum is the optimal solution for a finger switching.

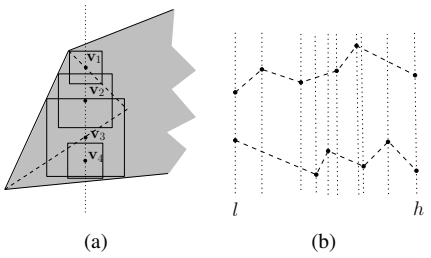

Fig. 2. The largest square on Voronoi edges

IX. Experimental Results

We have implemented the regrasp planning for a polygon with a large number of edges based on the switching graph concept. The program is written in C++ programming language. All run times are measured on a PC with a 2.4 GHz CPU.

Some test polygons with varying number of edges are shown in Fig. 3. We also vary the half-angle of the friction cone by $10°$, $15°$ and $20°$.

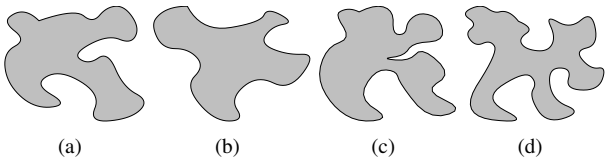

Fig. 3. Test polygons with number of edges (a) 128 (b) 200 (c) 256 (d) 300

TABLE I
SWITCHING GRAPH CONSTRUCTION OF $10°$ HALF-ANGLE

Fig.	#node	#edge	#CC	time for nodes	time for edges
(a)	70	162	15	0.11	0.03
(b)	35	79	6	0.20	0.03
(c)	84	222	11	0.25	0.05
(d)	134	400	11	0.30	0.09

The results of switching graph constructions are shown in Table I-III. They present for all test objects the number of connected polygons or nodes of switching graphs, edges of switching graphs, the number of connected components

TABLE II

SWITCHING GRAPH CONSTRUCTION OF 15° HALF-ANGLE

Fig.	#node	#edge	#CC	time for nodes	time for edges
(a)	65	194	5	0.16	0.03
(b)	41	121	4	0.22	0.02
(c)	85	323	3	0.31	0.06
(d)	150	630	4	0.39	0.19

TABLE III

SWITCHING GRAPH CONSTRUCTION OF 20° HALF-ANGLE

Fig.	#node	#edge	#CC	time for nodes	time for edges
(a)	58	230	3	0.17	0.03
(b)	41	156	2	0.27	0.03
(c)	84	424	2	0.38	0.06
(d)	143	782	2	0.45	0.08

of the switching graphs, time spent in node and edge construction in second. The number of nodes of a switching graph depends on the object's shape. An object with more complexity produces more connected polygons. The number of connected components indicates probability to have a path joining any two nodes in the switching graph. The half-angle of the friction cone heavily effects the results. It's clear that larger friction cone induces larger feasible grasp sets. This causes sets of force-closure grasps to be merged more and connected polygons to be larger. As a result, the number of nodes decreases while the number of edges increases so that the number of connected components of a switching graph decreases. Run times of node constructions depend on areas of merged connected polygons which are inherited from the objects' shapes and the values of the half-angles. For an edge construction, a run time relates to the number of connected polygons.

An example of a regrasp sequence is presented in Fig. 4. The sequence is computed using the algorithm described in Section VIII-A. Fig. 4(d) and (g) show regraspings by finger alignings. The dashed lines are lines connecting two contact points which entirely lie in the two associated friction cones.

X. CONCLUSION

We have proposed a method for solving the regrasp planning problem for a polygon with a large number of edges. The hand used in this work is assumed three free-flying fingers. Our method provides complete solutions represented by a graph which allows us to plan a regrasp sequence by using a graph search. The experimental results show the efficiency of our algorithm which merges grasp sets that are adjacent to one another into one connected set. The obtained connected sets are used to construct a switching graph in realtime. For our future works, we are interested in integrating constraint, such as hand kinematics and reachability, into our graph search so that a more practical sequence of regrasps can be obtained. Since an obtained result is a set of general solutions satisfying force-closure thus other constraints can be taken into account to determine an appropriate regrasp sequence for a given hand platform. We will also complete the implementation of the optimization of a regrasp sequence proposed in Section VIII-B.

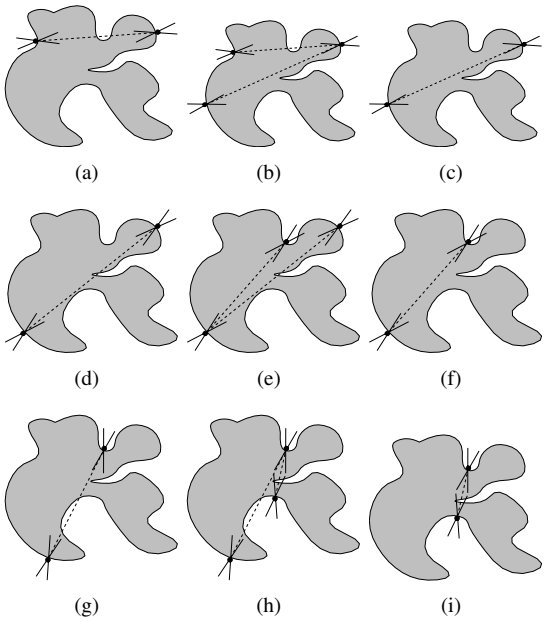

Fig. 4. A regrasp sequence for the object in Fig. 3(c) when the half-angle is 15°.

REFERENCES

[1] J. Hong, G. Lafferriere, B. Mishra, and X. Tang, "Fine manipulation with multifinger hand," in *IEEE Int. Conf. on Robotics and Automation*, 1990.

[2] D. Montana, "The kinematics of multi-fingered manipulation," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 4, pp. 491–503, 1995.

[3] J. Xu and Z. Li, "A kinematic model of finger gaits by multifingered hand as hybrid automaton," *Automation Science and Engineering, IEEE Transactions on*, vol. 5, no. 3, pp. 467–479, July 2008.

[4] A. Sudsang and T. Phoka, "Regrasp planning for a 4-fingered hand manipulating a polygon," in *IEEE Int. Conf. on Robotics and Automation*, 2003.

[5] M. Roa and R. Suarez, "Regrasp planning in the grasp space using independent regions," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009.

[6] V.-D. Nguyen, "Constructing force-closure grasps," *International Journal of Robotics Research*, vol. 7, no. 3, pp. 3–16, June 1988.

[7] J. Ponce and B. Faverjon, "On computing three-finger force-closure grasps of polygonal objects," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 6, pp. 868–881, December 1995.

[8] T. Phoka, P. Vongmasa, C. Nilwatchararang, P. Pipattanasomporn, and A. Sudsang, "Planning optimal independent contact regions for two-fingered force-closure grasp of a polygon," in *IEEE Int. Conf. on Robotics and Automation*, 2008.

[9] B. Faverjon and J. Ponce, "On computing two-finger force-closure grasps of curved 2D objects," in *IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, April 1991, pp. 424–429.

[10] E. Papadopoulou and D. T. Lee, "The l_∞-voronoi diagram of segments and vlsi applications." *Int. J. Comput. Geometry Appl.*, vol. 11, no. 5, pp. 503–528, 2001.